2003

A Web-based process and process models to find and deliver information to improve the quality of flight software

J. Scott Hawker

Hong Ma

Randy Smith

A WEB-BASED PROCESS AND PROCESS MODELS TO FIND AND DELIVER INFORMATION TO IMPROVE THE QUALITY OF FLIGHT SOFTWARE

J. Scott Hawker, Hong Ma, Randy K. Smith, University of Alabama, Tuscaloosa, Alabama

Abstract

Aerospace systems demand high-quality software engineering processes to deliver high-quality products. Although most aerospace organizations have high-quality processes, many of these processes fail to deliver to the engineer the organization's wealth of information and experience – information and experience that can further contribute to the quality of software products and engineering processes. In this paper, we present an interactive, web-based process support tool that delivers the information in a flight software engineering process as well as associated standards, lessons learned, and background information. The tool is based on an underlying formal model of the software engineering process activities and artifacts. This underlying model provides a semantic basis for context-based search and for reasoning about the engineering process. The result is an information portal to search for and deliver process and project-specific information to support the development of flight software.

Introduction

Aerospace systems demand high-quality software. A high-quality software engineering process helps assure the development of high-quality software products. Most organizations developing aerospace software have high-quality software engineering processes. However, many of these are incomplete in that there is a wealth of information and experience on software engineering techniques that can further contribute to the quality of the products and processes. This information is not available in the context of the process. In addition, many software engineering processes are difficult to use, and so they are not used as rigorously as they should be used.

In this paper, we describe an interactive, web-based software engineering process tool. The tool presents a software engineering process for the development of space flight software. The tool also links the process to associated techniques and background information, including organizational standards, best practices, lessons learned, and training materials. The tool supports the instantiation of the process for specific projects, allowing the process to be tailored for a project. The project-specific instance can be coupled with a workflow engine, allowing project-specific documents and products to be linked to the process.

To support the dynamic interaction and the integration with a workflow engine requires that the process tool be more than a set of web pages. The tool is based on a formal underlying model of the software engineering process activities, process information, and software artifacts in a project. The model enables additional capabilities, such as reasoning about the process and its execution, assembling process instances from reusable process components, and supporting activity context-specific search for additional information, such as relevant standards and lessons learned or relevant artifacts from prior projects.

The process tool, then, becomes an information portal, providing the software engineer with a wealth of general and project-specific information they need to follow their engineering process in practice. This paper describes the design of such an information portal and the underlying information models.

Initial Prototypes

We have focused our implementation prototypes on the Software Development Process Description (SDPD) [1] for a group that develops flight software. The SDPD defines their software engineering activities and procedures. We prototyped a web portal that presents standards and other information for use while performing SDPD-defined software design activities. We then researched a number of standards for modeling engineering processes and used their models as guidance toward a formal representation of engineering activities and documents used. We re-characterized the SDPD software design activities

0-7803-7844-X/03/$17.00 © 2003 IEEE

using this model, and we are now re-implementing the web portal prototype to be built on the underlying model. We are now also enhancing the model to capture the instantiation of the generic process for a specific project to build a specific flight software product, and to describe how a generic process might be a tailoring or modification of another process. The next sections describe this series of models and web portals and the use of standards to develop the models.

Description of an Activity

The SDPD describes the software development process as a collection of activities that develop software artifacts (requirements, designs, implementation units, test cases, etc.) or that support the engineering process (gathering metrics, contract monitoring, defining the process, etc). The activity definitions follow a regular pattern of description:

- Activity Name and Purpose
- Documents Used/Required
- Task Description
- Task Responsibility/Activities
- Review Process
- Product(s)/Document(s) Developed
- Tools
- Measures

Focusing on SDPD software design activities, we developed a web portal prototype to deliver information to activities with the above structure. The development of the initial SDPD Web Portal prototype used HTML and Javascript as "hard-coded" representations of the engineering activities and associated information.

Figure 1 shows a top-level view of the web portal. The content of the process is presented in the right-side pane. The user navigates the process via a tree navigation menu in the left-side pane or via web links embedded in the content of the right-side pane. Additional information not in the process, such as, the glossary, standards, lessons learned, document templates and examples, etc. are available via embedded links, links in the navigation tree, and links across the top of the page. Figure 2 shows more detail in the navigation tree and shows the content and links associated with the Preliminary Software Design task description.

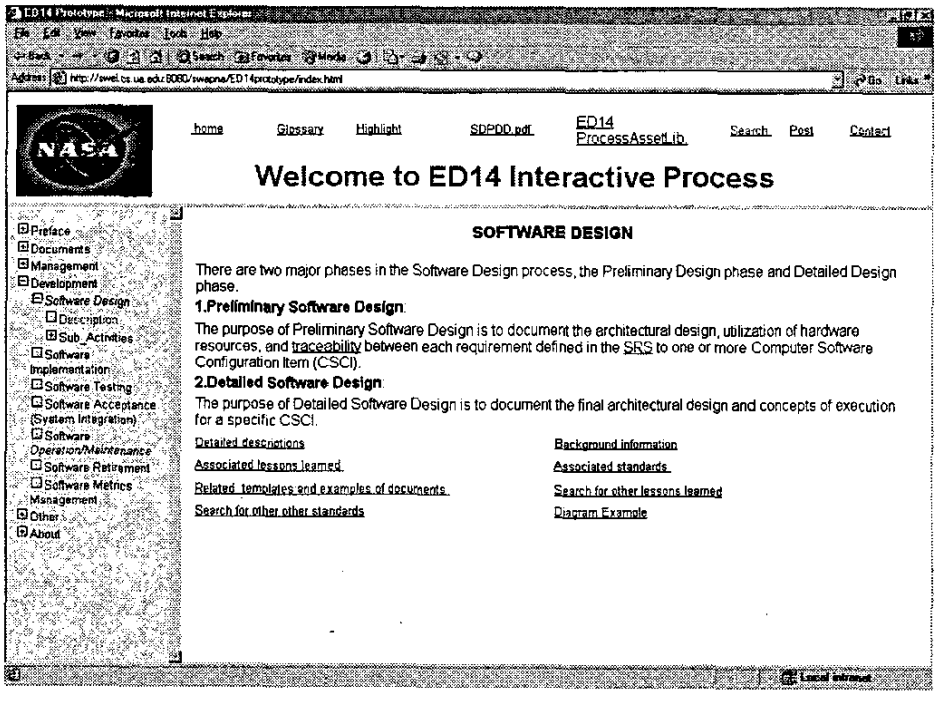

Figure 1. Web Portal Prototype Showing Process Content in the Right Pane, Process Navigation in the Left Pane, and Navigation to General Information in the Top Pane

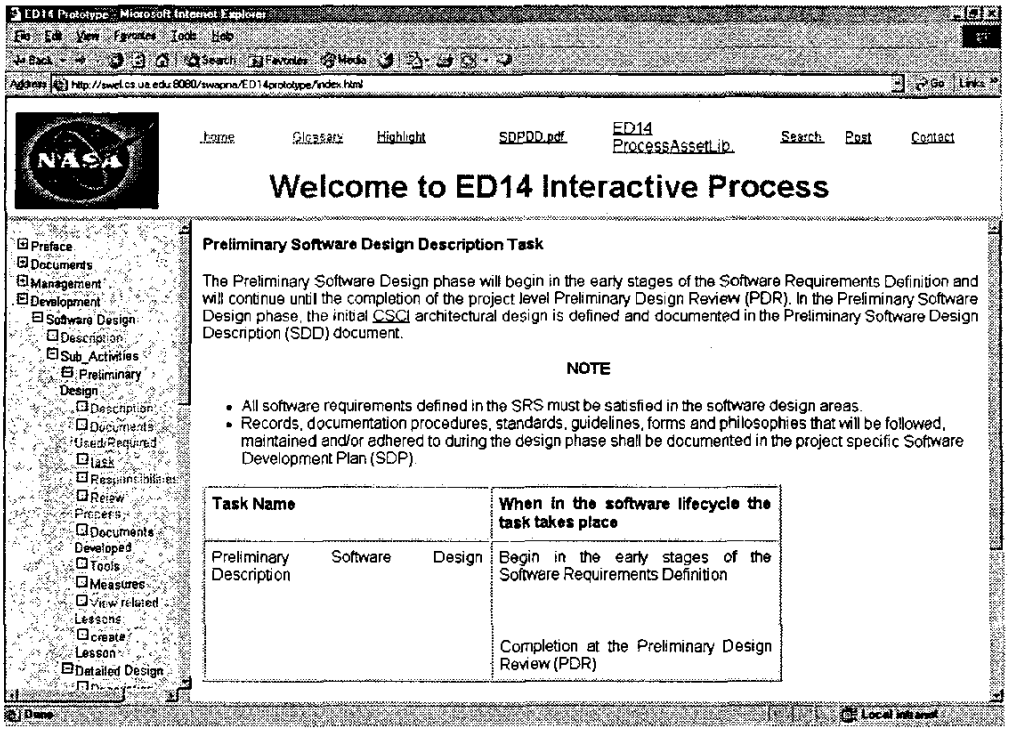

Figure 2. Web Portal Prototype Showing Details of Preliminary Software Design Activity

Activity Models – Formality

Underlying the user-visible web pages that deliver the process and associated information is a semantic model of the software engineering process. The web pages are a visualization of the underlying model of activities and work products. The underlying semantic model drives the user-interactive web pages, and software agents (such as search and workflow engines) that use the underlying model augment the user interaction.

The underlying model provides a foundation for a number of capabilities, including process automation (via integration with a workflow engine), dynamic assembly and tailoring of process elements from reusable process components, formal reasoning about activity specifications and reusable work products, semantic-based search for activity-relevant standards and lessons learned, assurance of process conformance, and reasoning about process usage metrics and process improvement opportunities.

UML as the Common Representation from Which All Others are Derived

We chose the Unified Modeling Language (UML) [2] as the basic knowledge representation language for the underlying model. UML provides a precise, yet understandable, way to represent and visualize knowledge. We use UML to provide the common abstract representation of knowledge. From these UML models, we derive numerous concrete representations that implement the models as computer data structures and programs. Specifically, our UML model identifies knowledge elements and types of elements (objects and classes in UML) and relationships between knowledge elements (associations in UML). Some of the models are then transformed to and implemented as Java and Perl language programs. Some of the models are transformed to and implemented as HTML files and Javascripts interpreted by a web browser. In a next iteration, the model elements also will be implemented as XML objects that are interpreted by XML Stylesheet definitions for user display and interaction, where the XML objects are

created and manipulated by Java, C#, and Perl programs.

By mapping alternate formats to the common, UML-based representation, we can translate between multiple concrete representations using the abstract UML-based representation as an intermediary. In this way, we can convert process knowledge into a format for use by a tool, viewer, or software agent available for that format.

Activity Model Standards

Research in the 1980s and 1990s resulted in numerous approaches to modeling processes and activities. Best practices in these areas were merged into industry consensus standards. The IDEF family of standards (we focus on IDEFØ [3]) are used to model manufacturing production systems and supporting information systems. The OMG Software Process Engineering Metamodel (SPEM) standard [4] is used to model software engineering processes as a configurable assembly of activities and work products. Business process modeling standards, such as the Workflow Management Coalition's reference models [5] and the associated OMG Workflow standard [6], focus on the information to support business activities (including engineering activities) and the flow of control between activities (workflow). These standards guide our modeling effort.

Process Definition, Enactment, and Execution

A process is a collection of activities executed and coordinated to achieve goals. Activities produce a result (output) by acting upon one or more inputs. Inputs and outputs are data and/or physical objects (materials, documents, etc.) involved in an activity. Inputs serve different roles in the activities. Some inputs are consumed or transformed to produce outputs, and some inputs are guidance or control on how an activity is performed. Associated with an activity is a resource or mechanism that performs the activity.

Figure 3 illustrates a graphical notation for an activity, where inputs are represented by arrows placed on the left side of an activity symbol, outputs are represented by arrows on the right, controls are arrows on the top, and mechanisms are arrows on the bottom (note: the meaning implied by the arrow placement is from the IDEFØ standard, and the Activity symbol is from SPEM). Figure 4 shows the Preliminary Software Design Activity in the SDPD. An activity graph connects outputs of activities to inputs of other activities. Figure 5 shows an example of an activity graph from the SDPD.

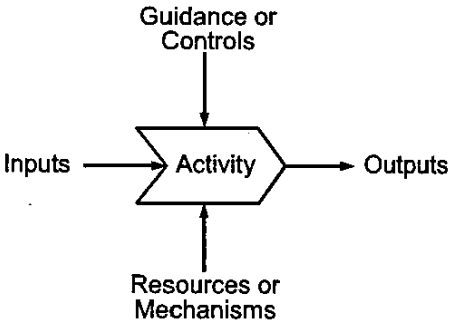

Figure 3. An Activity Model

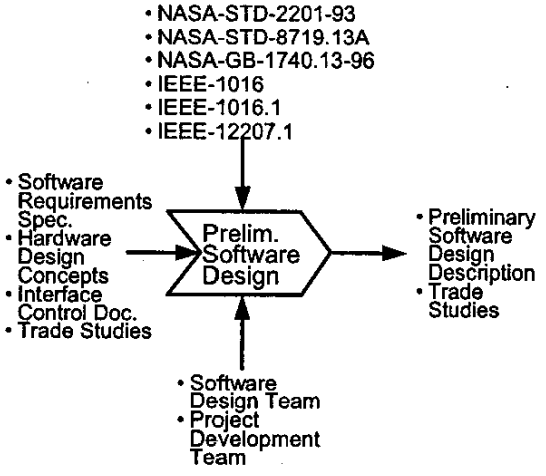

Figure 4. Inputs, Outputs, Guidance and Resources for the Preliminary Software Design Activity

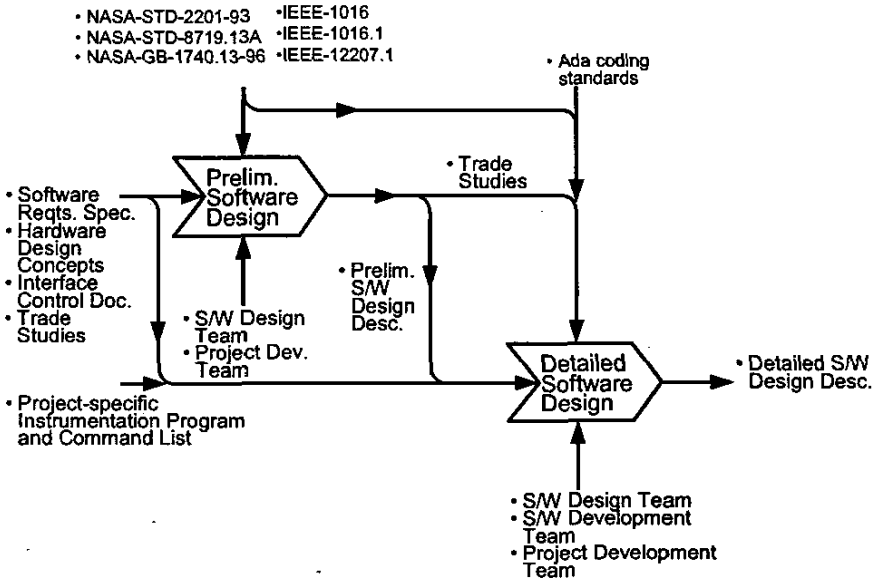

Figure 5. The Sub-Activities of the Software Design Activity

An activity is a generic definition of how to perform an engineering task. A generic activity is realized when specific instances of input data and objects become available for consumption, and the activity then produces specific output data and objects. The process of creating activity instances and controlling (coordinating and sequencing) their execution is called enactment. The enacted activities are executed by resources, including people (manual), people using tools (aided), or automation (automated). Figure 6 illustrates process enactment.

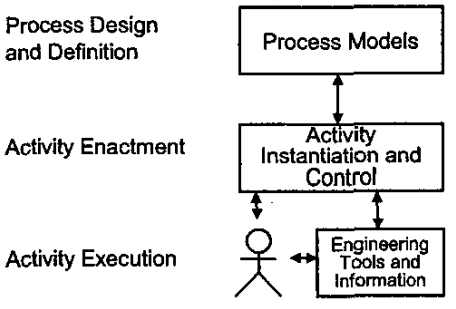

Figure 6. Process Definition, Enactment, and Execution (from [6])

UML Activity Model

Through several iterations of model refinement to incorporate SDPD process concepts and formal models from the modeling standards, we developed an activity model in UML. The model is organized into three packages, as Figure 7 shows. The next subsections detail the packages of Figure 7, then show how those package elements can be instantiated for the SDPD process.

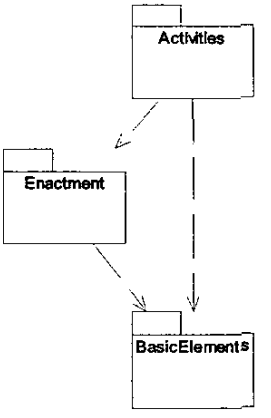

Figure 7. Packages in the Activity Model

Basic Elements Package

Figure 8 shows the elements in the Basic Elements package of Figure 7. It is derived from SPEM [4]. Every element in the model can have associated guidance. GuidanceKind categorizes types of guidance, including, for example, "technique", "guideline", "directive", "checklist",

"tool mentor", "template", "estimate", etc. Every model element also can have an external description, which provides for the linking of a model element to a document or other visualization for human use. Figure 9 summarizes the types of ModelElements in the model.

Enactment Package

Figure 10 shows the elements in the Enactment Package of Figure 7. This model is derived from the OMG Workflow Management Model [6]. It provides the mechanisms for an external workflow engine to start and complete a specific instance of an activity and to record history about that activity instance.

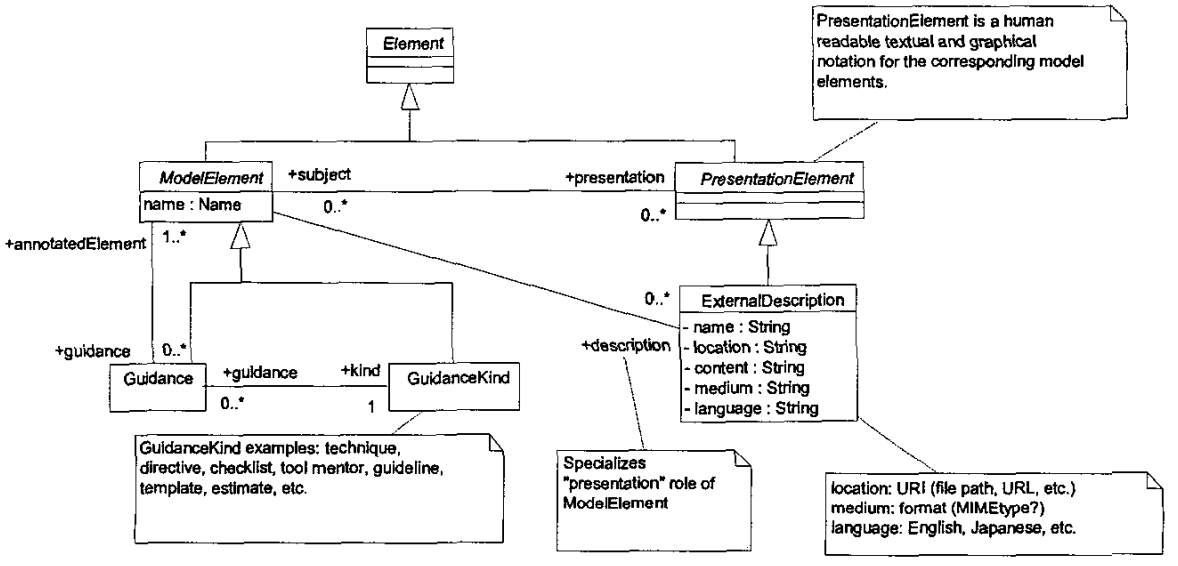

Figure 8. Basic Elements Package

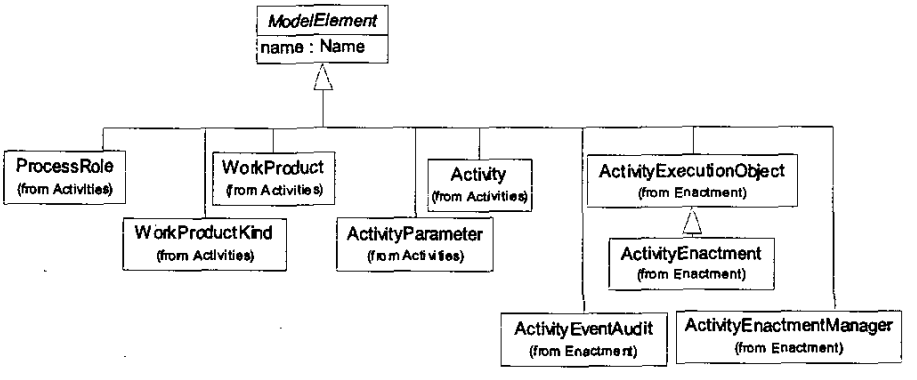

Figure 9. Top-Level Specialization Hierarchy

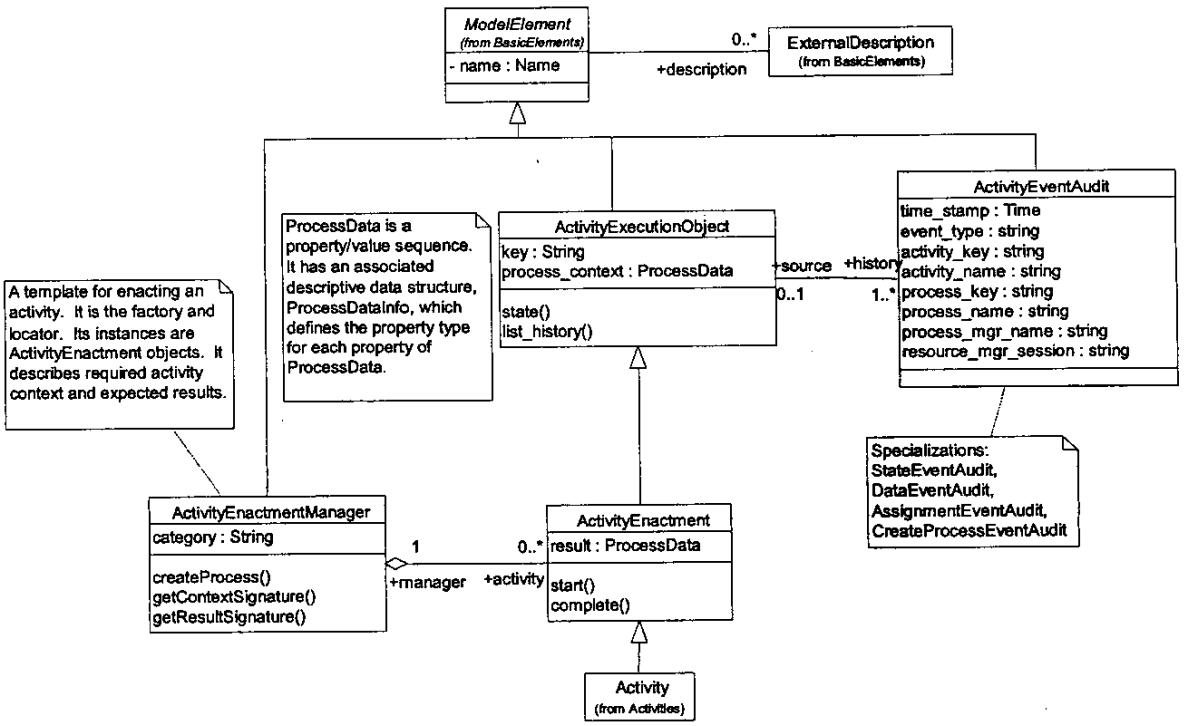

Figure 10. Enactment Package

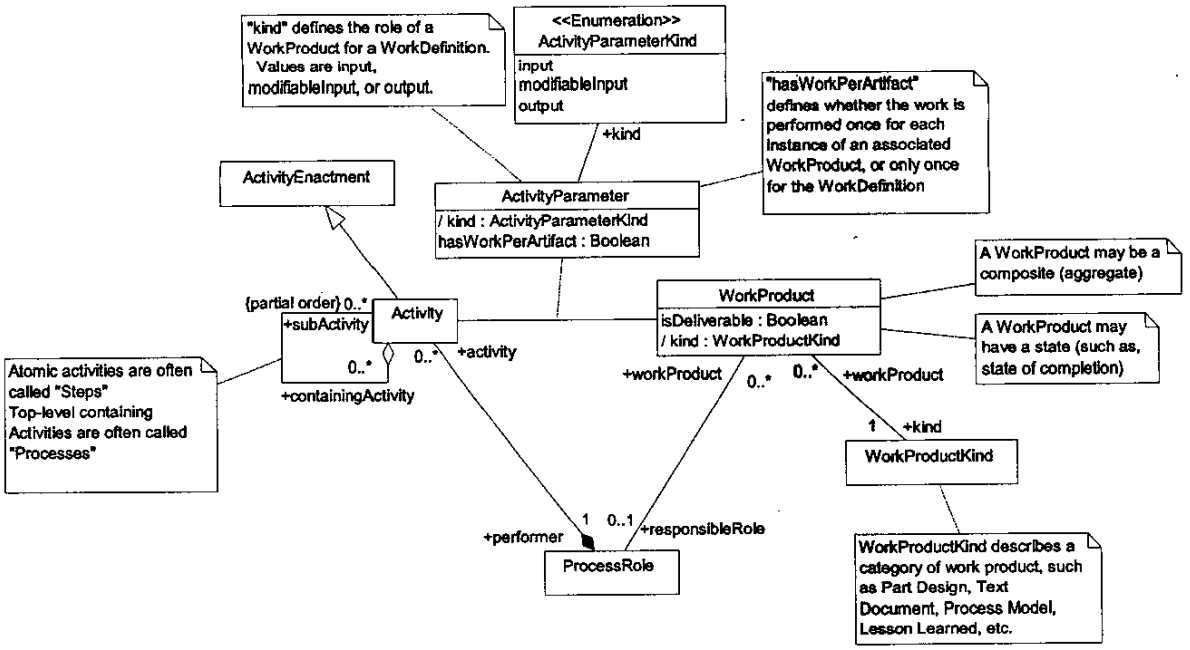

Figure 11. Activities Package

Activities Package

Figure 11 shows the elements in the Activities Package of Figure 7. This model is derived from the OMG SPEM standard [4]. It models a process role (implemented by a person and/or a software system) performing an activity that consumes input work products in creating output work products.

SDPD Web Portal, Revisited

Applying the UML activity model to the SDPD, we get the series of object (class instance) diagrams in Figure 12 through Figure 15. Note that these diagrams are not intended to be user interfaces for the web portal. Rather, they depict the software data structures – the data that will be transformed for display and will also serve as a knowledge base for search and reasoning. For search and reasoning, knowing the names, types, and relationships among the specific elements will be valuable. For example, the words in the name of an activity can be used as keywords in a search for information about those activities.

Figure 12 shows the activities and sub-activities associated with software design in the SDPD. Figure 13 captures alternative visualizations of the Preliminary Software Design activity. These visualizations are intended as graphical elements for a user interface (text, web page, or flow diagram) or for insertion in building a document (for example, automatically generate a process summary by inserting an activity name, its plaintext description, and its sub-activity names in a document). Figure 14 captures the input work products and output work products for the Preliminary Software Design activity. Note how the Trade Studies input work product is transformed by the activity into an updated output work product (it has ActivityParameterKind value of 'modifiableInput'). Figure 15 captures the standards and other guidance associated with the Preliminary Software Design activity.

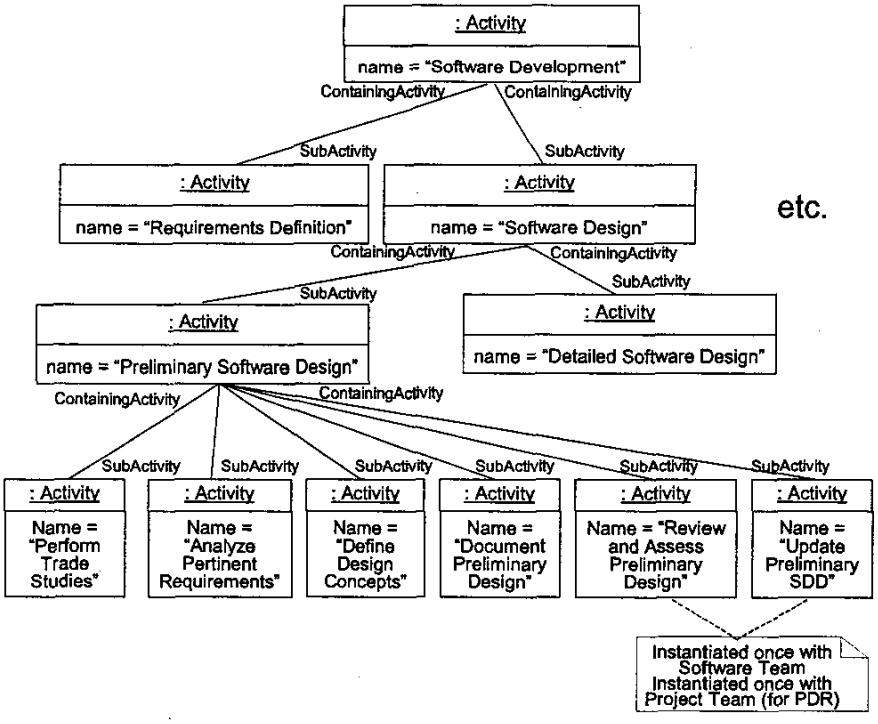

Figure 12. Activity Aggregation Hierarchy

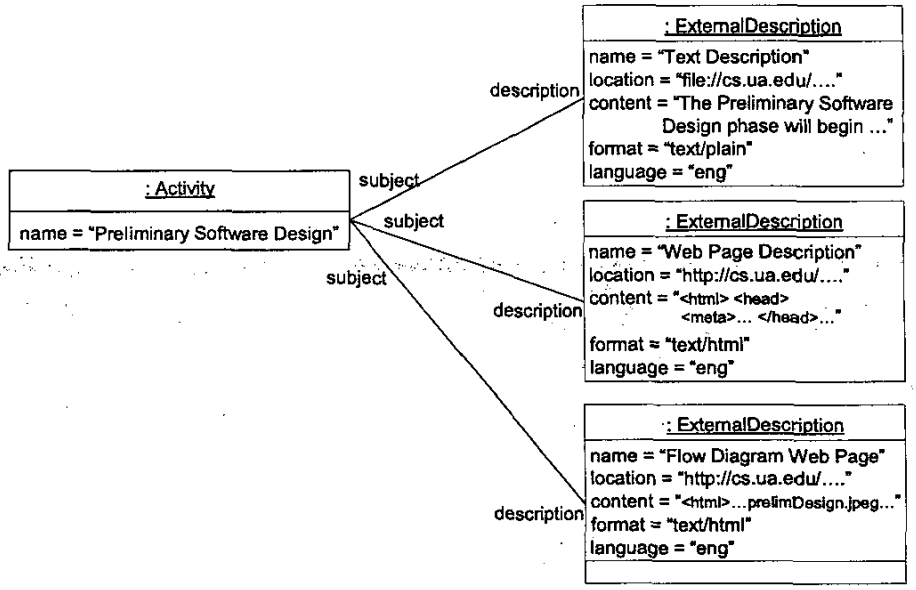

Figure 13. Activity Descriptions: Text, Web Page, and Flow Diagram

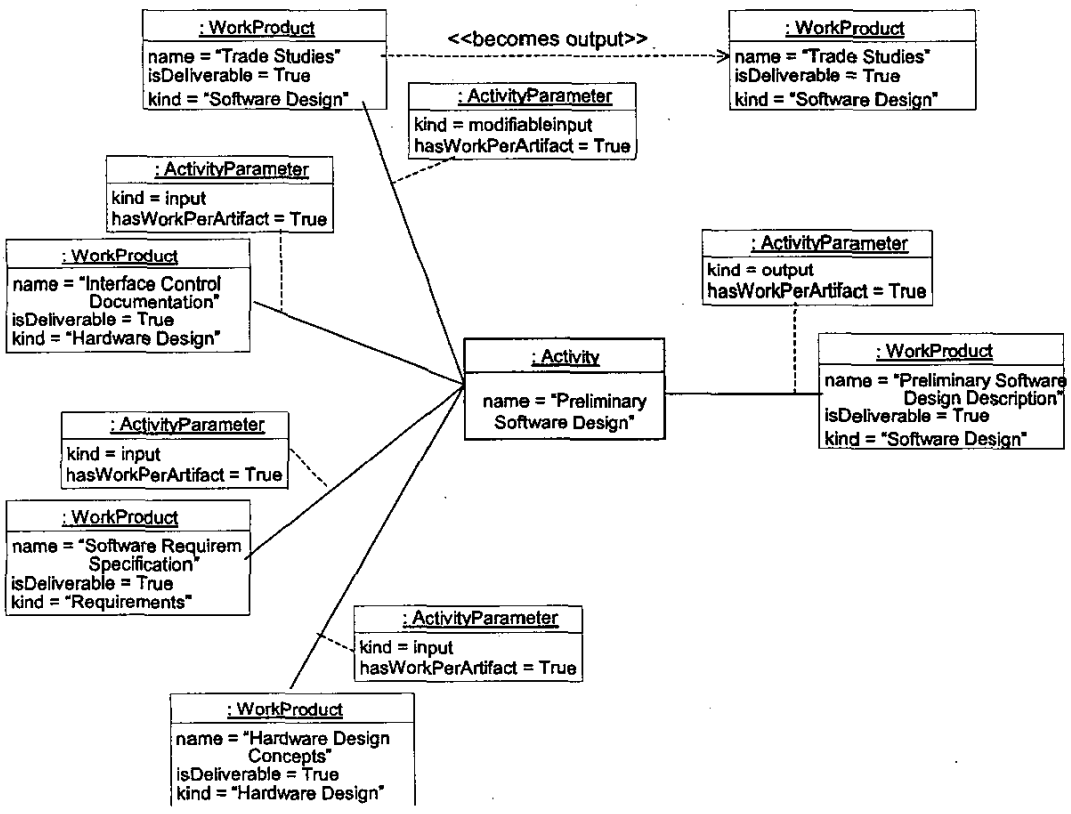

Figure 14. Input and Output WorkProducts for Preliminary Software Design

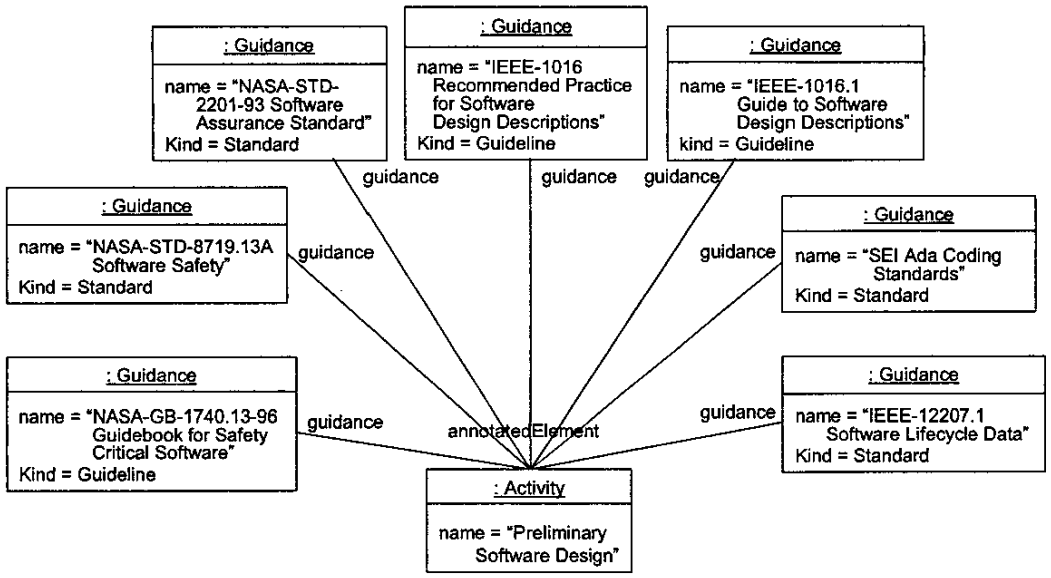

Figure 15. Guidance Elements for the ED14 Preliminary Software Design Activity

Next Steps

This section captures some possible next steps in developing and demonstrating the models and technologies for an SDPD web portal.

Data-Driven (Not Hard-Coded) Web Pages

Given the data structures (model class instances) capturing the software engineering activities, we could write scripts to generate web pages in the SDPD web portal. The pages could look the same as those in Figure 1 and Figure 2, except that now the pages would be data-driven – created dynamically from the underlying knowledge base – rather than hard-coded. We have not yet implemented this data-driven web page approach, but it is a necessary next step to validate our overall concept.

Capture Annotations and Lessons Learned

Future work includes tool support for capturing annotations and comments on the various model elements. This will be especially important in supporting the capture of lessons learned.

Process Tailoring, Instantiation, and Enactment

A planned next step is to develop a prototype of the process of instantiating the activities for a specific product development project, exercising the elements of the activity enactment portions of the model.

Further Towards an Ontology that Supports Reasoning

The development of the SDPD tool provides us an excellent opportunity to brainstorm and research the reasoning abilities we have enabled. The formal representation of the engineering activities and supporting information provides an exceptional framework to investigate the knowledge transfer and enabling information for varied engineering activities. We have mentioned numerous possibilities to explore.

We plan to map the activity model to the Process Specification Language (PSL), a draft standard for a formal, first-order logic ontology for engineering and manufacturing processes [7]. Mapping to PSL would give us a more carefully constructed model, it would give us more experience in the formal methods, and it would give us insight into how to use UML and other ontology-like languages, positioning us to adopt the Ontology

Web Language (OWL) [8] and Standard Upper Ontology [9] that are hoped to enable the next generation of semantic web technologies.

Capturing Software Engineering Standards and Other Information as a Knowledge Base

Future plans include building an ontology-based knowledge base that captures the software engineering activities in organizational standards and industry standards. Providing this consensus-based knowledge on software activities could provide a valuable asset to flight software development organizations.

The model-driven approach allows us to search for and integrate additional knowledge into the SDPD web portal. Development guidelines from the Software Engineering Institute, best practices, design patterns, design aides, training materials, and other information may help guide engineers toward quality products and processes.

Conclusion

We have described an interactive, web-based process support tool that presents the process information for a flight software development organization. We have begun to link additional information to the process through the web pages, including standards and lessons learned. More importantly, we have begun to develop a formal underlying model of the software engineering process. This model enables capabilities such as process automation, process assembly, and context-based search for additional information. Further prototyping effort will base the process support tool on the underlying model and will leverage the newly-enabled capabilities.

References

[1] NASA Marshall Space Flight Center Flight Software Group (ED14), May 2002 and May 2003, Software Development Process Description Document, Document ED14-SS-001, Revision J (May 2002) and L (May 2003), Marshall Space Flight Center, Huntsville, AL.

[2] Object Management Group, 2003, Unified Modeling Language (UML) Specification Version 1.5 (OMG document formal/2003-03-01), Framingham, MA, Object Management Group, http://www.omg.org/uml.

[3] SofTech, Inc., 1981, Integrated Computer-Aided Manufacturing (ICAM): Function Modeling Manual (IDEF0), F33615-78-C-5158, Wright-Patterson Air Force Base, OH, Materials Laboratory, Air Force Wright Aeronautical Laboratories. Available at http://www.idef.com/.

[4] Object Management Group, 2002, Software Process Engineering Metamodel (SPEM) Specification Version 1.0 (OMG document formal/2002-11-14), Framingham, MA, Object Management Group, http://www.omg.org.

[5] Hollingsworth, David, 1995, The Workflow Reference Model, Winchester, Hampshire, UK, Workflow Management Coalition, http://www.wfmc.org.

[6] Object Management Group, 2002, Workflow Management Facility Specification Version 1.2 (OMG document formal/2002-05-02), Framingham, MA, Object Management Group, http://www.omg.org.

[7] Schlenoff, Craig, Michael Gruninger, Florence Tissot, John Valois, Josh Lubell, Jintae Lee, 2000, The Process Specification Language (PSL): Overview and Version 1.0 Specification, NISTIR 6459, Gaithersburg, MD, National Institute of Standards and Technology, http://ats.nist.gov/psl/.

[8] McGuinness, Deborah and Frank van Harmelen editors, 2003, Web Ontology Language (OWL): Overview, Worldwide Web Consortium (W3C), http://www.w3.org/2001/sw/WebOnt/ .

[9] IEEE Standard Upper Ontology Working Group, IEEE P1600.1, http://suo.ieee.org.